Docket No.: 002950.P053
Express Mail No.: EM560646912US

UNITED STATES PATENT APPLICATION

FOR

SYSTEM AND METHOD FOR AUTOMATED AND CUSTOMIZABLE AGENT AVAILABILITY AND TASK ASSIGNMENT MANAGEMENT

Inventors:

Robert H. Joyce
Munisekaran Madhdhipatla
Allan Michael Moore
Rick A. Perotti

Prepared By:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard, 7th Floor
Los Angeles, California 90025-1026
(310) 207-3800

SYSTEM AND METHOD FOR AUTOMATED AND CUSTOMIZABLE AGENT AVAILABILITY AND TASK ASSIGNMENT MANAGEMENT

BACKGROUND

Field of the Invention

The invention relates to the field of customer service and the handling and processing of customer service requests. More specifically, the invention relates to handling and processing customer service requests received over a plurality of heterogeneous media.

Background

Customer service is an important part of any business. Many companies provide customer service via call centers in which incoming customer telephone calls are received and answered by customer service agents. Typically, each call center is staffed by agents with various skills. Calls are routed to an agent having an appropriate level of expertise for handling the particular kind of call. For example, an agent may have a language skill such as fluency in Spanish (in addition to English) or have expert knowledge of a particular product line.

As the internet has expanded, many companies are taking advantage of this communications medium to provide added methods of customer service. Many companies receive customer requests and comments sent by email either directly or through a web site. Such companies employ customer service agents to respond to these email messages. Responses may be in the form of an email reply or a telephone call.

In addition, the internet provides the ability to interactively service customers while the customer is accessing the company's web site. Such interactive customer service may take the form of interactive text, sometimes known as chat, or interactive voice over the internet. In the interactive voice scenario, a customer speaks into a microphone attached to the customer's computer while viewing the company's web site. The company must also

employ persons to handle such internet text and internet voice customer service requests.

BRIEF SUMMARY OF THE INVENTION

A system and method for blending tasks received from a plurality of media switches. The method comprises receiving a plurality of task data indicating a plurality of tasks and a plurality of agent data indicating a plurality of agents. The task data and the agent data are stored in a database system. Tasks are assigned to the agents according to workflows. The system comprises a blending engine coupled to a plurality of media switches and a plurality of agent workstations coupled to the blending engine. The blending engine receives a plurality of task data from the media switches. The agent workstations provide a plurality of agent data to the blending engine. The blending engine provides a plurality of task assignments to the agent workstations according to workflows.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a high-level schematic of the architecture of a blending system.

Figure 2 illustrates the flow of actions taken by a blending system upon receiving an incoming task.

Figures 3A and **3B** illustrate a more detailed schematic of the architecture of a blending system.

Figure 4 illustrates a more detailed the flow of actions taken by a blending system upon receiving an incoming task.

Figure 5 illustrates the flow of actions taken by a blending system upon an agent becoming available.

DETAILED DESCRIPTION

A. Overview

The invention provides a system and method for handling and processing customer service requests received over a plurality of heterogeneous media. Customer service requests are referred to herein as tasks. Generally, the invention provides a system and method for blending tasks received from a variety of media. In one embodiment, the system and method of this invention provide for the blending of tasks for agents within a customer relationship center. These tasks may be voice calls, email notes, web interactions and any other interaction that can be handled electronically. In one embodiment, the blending method makes task routing decisions based on the contact center's workload across all media. While individual systems handle the specific media items, the blending method provides a single point of reference for work allocation decisions.

B. A Blending System

Figure 1 shows a high-level schematic of the architecture of a blending system. In one embodiment, the blending system includes media switches 10, a blending database 12, a blending engine 14, agent desktops 16, a workflow server 18, and a customer database 20. The media switches include, but are not limited to, an automatic call distributor (ACD) 22, a web server 24 and an email server 26. Agent desktops may be, in one embodiment, computers such as desktop computers or, in another embodiment, networked terminals, and may, in either embodiment, be paired with a telephone or a phone implemented through software.

Figure 2 shows the flow of actions taken by a blending system upon receiving an incoming task. A task is a customer contact item such as a voice call, email message, or web-collaboration request such as text chat or internet interactive voice. Tasks are received by a media switch dedicated to handling contacts in that particular medium. After a task is received by a media switch, as shown in block 30, the media switch attempts to route that task to an available

agent, as shown in block 32. If the media switch succeeds in finding an available agent, the agent, known as a non-blended agent, services the task, as shown in block 34. If the media switch cannot successfully assign the task, it sends task information to an adapter as shown in block 36, and the adapter forwards the task information to the blending engine, as shown in block 38. The blending engine stores the task in the blending database, as shown in block 40, and executes a routing workflow in the workflow server, as shown in block 42.

The routing or "task queued" workflow executes business rules to determine whether a suitable agent is available to service the task, as shown in block 44. This decision may be based on information contained in a customer database as well as information in the blending database. In one embodiment, the business rules used to determine assignment and routing are built as part of the deployment of the system and are customizable. In one embodiment, these rules may, for example, involve finding the last agent that spoke to a particular customer and attempting to route the call to that agent. If that agent is unavailable, the skills of available agents could be assessed to establish which agent possesses the skills that most closely match the needed profile. The ethos of the blending system is to evaluate the current situation, and deliver to an agent the most appropriate task at that time. When a task is completed, the status of the system is re-evaluated and an "agent available" workflow is executed to determine which task is now most appropriate to assign to the available agent. In one embodiment, the blending database holds all the tasks across all the media, and, in this way, takes a "whole world" view of the workload of the system. In contrast, media switches only make decisions about their particular medium in isolation of the rest of the system.

The decision of whether there is an agent available is passed to the blending engine from the workflow server. If no suitable agent is available, the task remains in the database as shown in block 46. If an agent is found to handle the task, the blending engine signals the agent's desktop application, which retrieves the task from the media switch, as shown in block 48. The agent

receives notification of the task assignment, as shown in block 50, and the blending engine determines whether the agent accepts the task, as shown in block 52. In one embodiment, the desktop application asks the agent to accept the task. In this embodiment, the agent may send a message to the blending engine via the desktop application declining the task. In one embodiment, the blending engine may time out if a task acceptance notification is not received in a specified period of time. If the agent accepts the task, the desktop application signals the blending engine that the agent accepted the task, as shown in block 54. In one embodiment, the blending engine does not respond to the media switch after receiving task data. The desktop application then requests the task from the appropriate media switch on behalf of the agent, as shown in block 58. The agent may then service the task, as shown in block 60. If when the agent receives the task assignment from the blending engine, the agent rejects the task, the desktop application signals the blending engine that the agent rejected the task assignment, as shown in block 56.

In the situation where no available agent is found, the task remains in the database. When an agent via the desktop application signals that the agent is available to take more work, the blending engine causes a workflow to execute. In one embodiment, an "agent available" workflow uses the agent skill information stored in the blending database to determine the most appropriate task for the agent to take at the time. This decision is passed back to the agent desktop via the blending engine, and the desktop application retrieves the task from the media switch.

In another embodiment, a "check system status" workflow may execute to monitor the status of the entire system. Execution of a "check system status" workflow may be event based or time based. That is, execution may be triggered upon the expiration of a defined period of time, such as, for example, every minute, and upon the occurrence of particular defined events, such as, for example, when a task is queued, or when a defined number of tasks are queued. In one embodiment, agents may have a status of "available if needed." In this

embodiment, the "check system status" workflow may call in agents when the system load exceeds a predetermined threshold by transitioning the status to "available" which causes the "agent available" workflow to run. In another embodiment, the "check system status" workflow interrupts agents, requesting that they abandon one task for another that has become more important. This embodiment is particularly effective for re-assigning agents handling email when call volumes suddenly rise. In this embodiment, the blending engine instructs agents to abandon their email tasks and switch to taking calls. In a related embodiment, the agent status may be set to AVAILABLE, UNAVAILABLE, or AVAILABLE IF NEEDED. In such an embodiment, the "check system status" workflow may assign tasks to agents having the status of AVAILABLE IF NEEDED if certain criteria are met. This embodiment is particularly useful if persons with other positions in the company have the skills needed to assist in handling tasks when the volume of tasks exceeds a predetermined threshold.

C. Blending System Architecture

1. Overall Functionality

Figures 3A and **3B** illustrate a more detailed schematic of the architecture of a blending system. (**Figures 3A** and **3B** combine to form one larger figure and should be considered together.) Blending engine 60 consists of four components: adapter manager 62, workflow broker 64, database manager 66, and agent manager 68. In one embodiment adapter manager 62 communicates with other components via adapter message handlers 90, and agent manager 68 communicates with other components via agent message handlers 92. More specifically, adapter manager 62 provides common interface connectivity between media switches 70 and blending engine 60. Workflow broker 64 is a conduit for invoking and receiving responses from workflow server 65. Database manager 66 provides connectivity to blending database 72. In one embodiment, the blending database is comprised of dynamic database 76 and a static database 78. Agent manager 68 manages communication between blending

engine 60 and agent desktop 74. The media switches 70 communicate with the blending engine via adapters 82 specifically designed for each particular switch. Each adapter 82 places messages received from a particular media switch 70 into a new message in common message format to be forwarded to blending engine 60. The common message format used in one embodiment is extensible markup language, more commonly known as XML, ver. 1.0 (more information is available from the World Wide Web Consortium, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139). In one embodiment, the interface between adapters 82 and blending engine 60 is provided via adapter queues 88 that allows for asynchronous communication. In one embodiment, adapter queues 88 are implemented as Microsoft Message Queue (MSMQ). More information on MSMQ is available from Microsoft Corporation, Redmond, Washington.

At least three messages are produced by each of adapters 82: task queued, task dequeued, and reset. More specifically, a "task queued" message is sent when a task needs to be assigned to an agent. A "task dequeued" message is sent when a task has been assigned to an agent, whether through blending or independently. In addition, a "task dequeued" message may be sent when the task has been abandoned such as, for example, when a caller hangs up. A "reset" message is sent to clear queues when an adapter is brought up and taken down. Other messages may be produced by adapters, including a "task queued" message.

Adapter manager 62 determines the medium to which the received message is related and passes that message to the corresponding adapter message handler 90. There is a one-to-one correlation between each of adapters 82 and each of adapter message handlers 90. Adapter message handlers 90 extract information from the common message format message, and perform actions relevant to the particular message type. That is, if the message signals that a task has been queued, the appropriate adapter message handler 90 will request database manager 66 to store the task in the relevant virtual queue within dynamic database 76, and will then request workflow broker 64 to initiate a "task

queued" workflow. When a response is received from workflow server 65, workflow broker 64 passes the response to an adapter message handler 90.

If an agent has been found to handle the task, adapter message handler 90 informs the agent manager 68 via the agent message handler 92 to pass the task data to the relevant agent desktop 74. Agent manager 68 sends a common format message (in one embodiment, an XML message) to desktop helper 94 via agent queue 96 (in one embodiment, an MSMQ) from which the desktop application 98 determines the medium and any other relevant information (such as call/email ID). Desktop helper 94 provides a communication layer between desktop application 98 and agent manager 68. In a similar way to the adapter helpers 83, desktop helper 94 manages agent queue 96 while the desktop application 98 creates and processes messages to be passed through agent queue 96. In addition, desktop helper 94 creates two messages: "connect" and "disconnect." The messages are passed into desktop helper 94 in a common message format, such as, in one embodiment, XML, and are forwarded to agent manager 68. Desktop application 98 uses the task data to select the correct media switch 70 with which to work and, via the relevant application program interface (API) 99, requests that the task be delivered to that agent. In one embodiment, when the desktop application is executed, connections to each of the media switches for which the agent is qualified are automatically established via the appropriate APIs.

If an agent is not found, the task will remain in the database until such a time as the media switch assigns it to a non-blended agent, or a blended agent becomes available. In the latter scenario, desktop application 98 sends a common format message (in one embodiment, an XML message) to desktop helper 94 to indicate that the agent requires more work. Desktop helper 94 forwards this message to agent manager 68 via agent queue 96, which then passes the request to workflow broker 64 via agent message handler 92. The flow from this point is the same as for task queued.

2. Media Switches

Media switches 70 receive, queue and deliver its associated medium in electronic format. In addition, media switches generate events signaling that a task has been queued, de-queued and, in the case of real-time interactions, disconnected. A media switch provides an interface to enable a desktop application to request and be delivered a task. Some media switches allow for specific tasks to be requested, while other media switches only provide the oldest, or other switch-determined, task. Media switches may be smart or dumb. Examples of media switches 70 include, but are not limited to, ACD 70A (such as those available from Aspect Communications of San Jose, California, Lucent Technologies of Murray Hill, New Jersey, and Nortel Networks Corporation, Brampton, Ontario, Canada) email server 70C (such as Aspect's Aspect Customer Email available from Aspect Communications of San Jose, California), and web server 70B (such as Web Interaction available from Aspect Communications of San Jose, California).

ACD 70A provides access to incoming telephone calls from a public switch telephony network (PSTN) 75. Live telephone calls are queued, and information about the calls is then routed to blending engine 60 when a non-blended agent is unavailable to process the incoming call. In another embodiment, voice mail may be stored by ACD 70A and then routed to blending engine 60 for processing.

A POP3 mail server 84 may also be a media switch. POP3 is short for Post Office Protocol, Version 3, an internet mail standard promulgated by and available from the Network Working Group of the Internet Engineering Task Force at http://www.ietf.org/rfc/rfc1939.txt. POP3 servers support receiving, queuing and delivering messages in email format. Although, POP3 servers do not have the capability to pro-actively notify an external system of the queued of an email message, in one embodiment, a POP3 adapter, effectively a wrapper application, is used to provide a simple polling mechanism to check the current state of the POP3 queue, thus giving a POP3 server full media switch capabilities.

Sub A1

In one embodiment, web interaction, text chat and voice over the internet capabilities are provided by web server 70B. Communication between an agent and a customer are provided by any method known to these skilled in the art to establish text chat and voice over internet connections. In such an embodiment, the customer's display may automatically mimic the sequences of internet web pages traversed by the agent. In such an embodiment, agent desktop 74 signals web server 70B via computer telephony integration (CTI) server 97 that a connection has been established, which is in turn picked up by CTI server 71. Web server adapter 82B, coupled to CTI server 71, translates establishment of a web server connection into a "dequeued" message and forwards the message to adapter manager 62 of blending engine 60 via adapter queue 88. Various other implementations are possible and readily known by those skilled in the art.

In another embodiment, each media switch includes a private database for storing configuration data. In one embodiment, whenever a media switch updates its database, the media switch sends a message to the blending database via the adapter manager of the blending engine which then updates the configuration data. In a related embodiment, the blending database may regularly poll the media switch databases for updated or new configuration data. Configuration data may include information such as, for example, a group data specifying to which group an agent belongs.

3. Adapters

a. Adapter

The role of the adapter is to provide connectivity to media switches. Each media switch 70 has a differing output in terms of the number of event messages it produces, and the format and content of those messages. To ensure that the blending engine 60 does not have to support multiple types of messages, adapters 82 offers a common set of messages across the whole range of switches. That is, ACD adapter 82A receives ACD specific messages from ACD 70A via CTI server 73 and processes them into common messages in a common message format; web server adapter 82B receives web server specific messages from web server

70B via CTI server 71 and processes them into common messages in a common message format; and email adapter 82C receives email specific messages from email server 70C via email queue 85 and processes them into common messages in a common message format.

In one embodiment, at least three common messages are supported: "task queued", "task dequeued" and "reset". A "task queued" message is sent when a task needs to be assigned to an agent. A "task queued" message causes the blending engine to add the task to the appropriate queue in the blending database and to invoke the "task queued" workflow to determine if an appropriate agent is available to service the task. A "task dequeued" message is sent when a task has been assigned to an agent. A "task dequeued" message causes the blending engine to remove the task from the appropriate queue in the blending database. A "reset" message is sent when an adapter has been brought up or taken down. The "reset" message causes the blending engine to empty the particular adapter queue 88 for the particular media switch.

Although these kinds of messages are common across adapters, in one embodiment, each message is named differently for each adapter so that the messages from different media switches are easily and instantly distinguishable and may be processed by the blending engine based on their name. In addition, even though the same kinds of messages may be used, the content of the same kind of message may vary for each adapter, even if the adapter is of the same type of adapter. For example, a task queued message from an Aspect ACD contain different information and different fields than a task queued message from a Lucent ACD.

In one embodiment, these common kinds of messages are formatted and transferred in a common message format such as XML. This offers the advantage that the blending engine has only one message format to process, while the actual structure and content of the messages is not restricted using a common message format also facilitates support for new and additional switches. Each media switch has different data pertinent to its particular medium. For

example, dialed number identification service (DNIS) and automatic number identification (ANI) services that provide the telephone number of a call, are important to a voice telephone call, but are not available or relevant to an email message; whereas email messages have to/from addresses and a subject line not available for voice telephone calls.

Although the internal format of the messages are different for each media switch, in one embodiment, the messages are all sent via XML. The adapter for a media switch takes the information produced by that switch and bundles it into a common message format, in one embodiment, XML message. The messages contain data items expected by the corresponding adapter message handlers 90 for the media switch. The common message format serves effectively as a protocol between the adapters 82 and the adapter message handlers 90. When support for a new media switch is required, a new adapter that produces common messages in the common message format must be created. The following is an example "call queued" message from an ACD adapter in XML format:

```
<?xml version="1.0" ?>
<START>
    <TYPE>
        ASPT_CALL_ARR
    </TYPE>
    <CUSTOM>
        <MEDIA_ID TYPE ="INT">
            5
        </MEDIA_ID>
        <TASK_ID TYPE ="INT">
            1029
        </TASK_ID>
        <APPLICATION_ID TYPE ="INT">
            126
        </APPLICATION_ID>
```

```
                <TRACK_NUM TYPE ="INT">
                    10529
                </TRACK_NUM>
                <DNIS TYPE="STRING">
                    18003254958
                </DNIS>
                <ANI TYPE="STRING">
                    4082651093
                </ANI>
                <STATUS TYPE="STRING">
                    ARRIVED
                </STATUS>
                <STATUS_CHANGE TYPE="TIMESTAMP">
                    1999-07-27 10:23:56.000100
                </STATUS_CHANGE>
            </CUSTOM>
        </START>
```

Adapter message handler 90 receives and parses this message, interrogates the value of TYPE, and then coordinates the processing required by that type of message. In this example, this includes instructing database manager 66 to add this information to blending database 72. Other messages, such as "call queued", may, in one embodiment, cause workflow broker 64 to sennd a message via workflow queue 67 to workflow server 65 which causes a workflow to execute. In another embodiment, a "call queued" or other message may cause adapter message handler 90 to instruct workflow broker 64 to send a "user defined event" to workflow server 65.

In one embodiment, when using XML the order in which the data items (contained in the <CUSTOM> ... </CUSTOM> tags) are positioned is unimportant. In this embodiment, if a data item is non-mandatory, it does not

need to appear at all. That is, the data item does not require a placeholder. The message contains information signaling that an event has occurred creating a task; the message does not contain the actual task itself. This is obvious for voice calls, but not so for email or other text-based media. The messages contain information about the task needed to make decisions about retrieving and processing the task.

b. Adapters and Adapter Helpers

A key role of adapters 82 is to translate native messages from the media switches into a common message format such as, in one embodiment, XML. Although there is a common message format, what is common about the format is the general structure and arrangement of the message. The internal information within each common message format message remains media specific. In one embodiment, the adapters 82 communicate with the adapter manager 62 via the relevant adapter queue 88. Whereas each adapter 82 is specific to each media switch 70, the communication between adapters 82 and the adapter manager 62 is generic. The adapter helpers 83 assist the adapter 82 by establishing a connection to adapter manager 62 on startup, closing the connection on shutdown, and managing messages on the adapter queues 88.

4. Blending Engine

a. Adapter Manager

Adapter manager 62 acts as the communication hub between adapters 82 and adapter message handlers 90. In one embodiment, when an adapter 82 has a message for the blending engine 60, the adapter places the message in a common message format. In one embodiment, to achieve this, adapter 82 wraps the message in XML and places it in an adapter queue 88. In one embodiment, a mandatory field in the common message format and, in one embodiment, in the XML message, is TYPE. This field allows the adapter manager 62 to determine to which adapter message handler 90 the message should be sent.

b. Adapter Message Handlers

The role of the adapter message handlers 90 is to extract the information sent by an adapter and act upon that information within the blending engine on the adapter's behalf. In one embodiment, there is one adapter message handler for each media switch. The adapter message handlers 90 serve three purposes. The adapter message handlers 90: (1) keep task information in the blending database 72 current; (2) invoke "task queued" workflows; and (3) request agent manager 68 to assign tasks to agents. Messages are passed to the adapter message handlers 90 via adapter manager 62 in blending engine 60. When the adapter message handlers 90 start up, each adapter message handler registers with the adapter manager 62 and specifies which kind of messages it processes. This registration allows for matching those messages produced by the corresponding adapter 82 to the appropriate adapter message handler 90.

c. Workflow Broker

Workflow broker 64 receives common message format messages from the adapter message handlers 90 to process "task queued" events. Workflow broker 64 receives common message format messages from agent message handler 92 to execute "agent available" workflow functionality. The content of these messages determines what event and parameters workflow broker 64 sends to workflow server 65 to cause the correct workflow to execute. In one embodiment, workflow broker 64 communicates with workflow server 65 via workflow queue 67. Communicating through a queue avoids dropped connections at high throughput. In one embodiment, workflow queue 67 is an MSMQ. In this embodiment, workflow broker 64 uses a Component Object Model Intelligent Route Administrator (COMIRA) component 108 to manage this communication. This is based on the Component Object Model (COM) standard popularized by Microsoft Corporation of Redmond, Washington.

d. Database Manager

Database manager 66 provides Open Data Base Connectivity (ODBC) communication between blending engine 60 and blending database 72. ODBC is

a standard database access method popularized by Microsoft Corporation of Redmond, Washington. ODBC uses Structured Query Language (SQL) as its database access language . In one embodiment, database manager 66 does not provide access to external databases, such as a customer database. In one embodiment, database manager 66 receives common message format messages from adapter message handlers 90 and the agent message handler 92. In some embodiments, these message are formatted as XML messages. In such an embodiment, these messages contain structured data informing database manager 66 which database, object and attributes need to be added to, updated or deleted from blending database 72. For example, a "call queued" message received by the ACD message handler 90 by the adapter manager 62 via an adapter queue 88 is bundled as an XML message and is sent to and informs the database manager 66 how to store that information in blending database 72. In this embodiment, the common message format XML message sent from the message handler to database manager 66 is:

```
<?xml version="1.0" ?>
<START>
    <TYPE>
        ASPT_CALL_ARR
    </TYPE>
    <HEADER>
        <STORAGE>
            DYNAMIC
        </STORAGE>
        <OBJECT>
            T_ASPT_CALL
        </OBJECT>
        <OPERATION>
            ADD
        </OPERATION>
```

```
</HEADER>
<DATA>
    <T_ASPT_CALL>
        <MEDIA_ID TYPE ="INT">
            5
        </MEDIA_ID>
        <TASK_ID TYPE ="INT">
            1029
        </TASK_ID>
        <APPLICATION_ID TYPE ="INT">
            126
        </APPLICATION_ID>
        <TRACK_NUM TYPE ="INT">
            10529
        </TRACK_NUM>
        <DNIS TYPE="STRING">
            18003254958
        </DNIS>
        <ANI TYPE="STRING">
            4082651093
        </ANI>
        <STATUS TYPE="STRING">
            ARRIVED
        </STATUS>
        <STATUS_CHANGE TYPE="TIMESTAMP">
            1999-07-27 10:23:56.000100
        </STATUS_CHANGE>
    </T_ASPT_CALL>
</DATA>
</START>
```

This XML message instructs database manager 66 to add data to the T_ASPT_CALL table in dynamic database 76. The data itself is held in the <DATA> ... </DATA> tag pairing. Database manager 66 parses this data, constructs the correct SQL statement to perform the action, and sends it to dynamic database 76 via ODBC.

e. Agent Manager

At startup, desktop helper 94 communicates with agent manager 68 to specify which queue the agent manager should use to send messages to that particular agent desktop. In one embodiment, the agent's ID is used as part of the queue name. In this way, agent manager 68 is able to distinguish between, and deliver tasks to the appropriate agent queue 96 and thus to the appropriate individual agent desktops 74. Like the communication between adapters 82 and adapter manager 62, there is one queue, agent queue 96, dynamically allocated for sending messages between agent manager 68 and each of the desktop helpers in a plurality of agent desktops (although only one agent queue 96 and one agent desktop 74 with one desktop helper 94 are depicted).

5. Blending Database

Blending database 72 holds configuration information about agents and media switches, and a copy of the queues for all media switches. The blending database is accessed by the various workflows to accomplish task and agent assignments. In one embodiment, the blending database comprises two databases, dynamic database 76 and static database 78. In one embodiment, the dynamic, in-memory database is a SQL database such as the TimesTen™ database system available from TimesTen Performance Software, Inc., 2085 Landings Drive, Mountain View, California 94043. Dynamic database 76 is a real-time in-memory database containing a virtual model of the queues on media switches 70. Blending database 72 also contains configuration information for speed of access. Using a dynamic, in-memory database as part of the blending database allows for fast access to information. In one embodiment, persistent information

such as agent and media configuration is stored in static database 78. In one embodiment, static database 78 is a writeable media database such as Oracle8i available from Oracle Corporation, 500 Oracle Parkway, Redwood Shores, California 94065. This static, writeable media database is used in conjunction with the dynamic, in-memory database to prevent data loss when a machine crashes.

Dynamic database 76 is, in one embodiment, an in-memory relational database management system (RDBMS) that is accessible through SQL via ODBC. The dynamic RDBMS has advantages over static RDBMS systems in that optimization algorithms can be more precise since they do not need to account for disk access. In addition, delays incurred while data is accessed from physical disks are eliminated when using a dynamic RDBMS. The dynamic, in-memory RDBMS is, therefore, significantly faster than traditional static RDBMS technologies.

Blending database 72 contains a number of standard entries. These entries may reflect, in various embodiments, multiple skill classifications of agents which are used by the workflows to assign tasks to agents. These entries may be, maintained by an admin client 112 that communicates with blending database 72 via database manager 66. If additional information is required to be stored in blending database 72, this additional information may be added according methods known to those skilled in the art. For example, a "first language" property may be added to media blending agent details. In this way, admin client 112 may be used to maintain blending database 72.

In one embodiment, an historical database is also used. Historical database 84 effectively listens in on what is occurring in the blending engine 60 via feed control 87. In one embodiment, feed control 87 receives information from all components of the blending system. For clarity, input to feed control 87 from adapter manager 62, workflow broker 64, database manager 68, agent manager 68, adapter message handlers 90 and agent message handlers 92 are not depicted. Historical database 84 is coupled to feed control 87 via historical

database queue 86. In one embodiment, historical database queue 86 may be an MSMQ. Historical database 84 receives XML formatted messages and may maintain historical information gathered from messages sent within the blending system. The admin client 112 may use the historical information to prepare useful information, such as, for example, how long it takes agents in general to handle particular kinds of tasks, how long it takes agents a particular agent to handle particular kinds of tasks, and which agents worked with particular client companies or particular persons in servicing prior tasks. Such information is useful to a customer service center administrator in developing customer service policies for agents to follow and managing employee agents.

6. Executive Service

The executive service 114 starts up and gracefully shuts down the blending engine components. When the executive service starts up, it enumerates all the managers and helpers, allowing the handlers to register with the managers, and the managers to register with each other.

D. <u>Workflows</u>

Workflows implement various business rules which vary depending on the particular embodiment and implementation. In one embodiment, workflows work to allocate agents and tasks from all media switches. In another embodiment, workflows only allocate tasks received over one kind of media switch. In one embodiment, the business rules invoked by the blending engine handle two scenarios: (1) finding an available agent to deal with an incoming task, processed by the "task queued" workflow; and (2) finding a queued task for an agent becoming available and requesting more work, processed by the "agent available" workflow. The latter is more likely to result in a successful assignment as it is far more common in contact centers for tasks to be queuing, waiting for an agent to become available, than for agents to be waiting for long periods of time for work to arrive. In one embodiment, a "check system status" workflow is also provided to periodically execute and achieve agent assignments or re-assignments when certain system conditions exist, such as if the system

volume is beyond a certain predetermined threshold. Workflow client 77 provides access to the blending system such that a system manager may amend existing workflows such as for customization. In addition, the workflow client allows for and provides an extensive system for adding new workflows beyond those discussed herein.

1. Task Queued Workflow

Figure 4 illustrates a more detailed the flow of actions taken by a blending system upon receiving an incoming task. Whenever a task is queued on a media switch, as shown in block 160, the media switch converts the task information into a common message format the blending engine can process, as shown in block 162. The media switch signals the blending engine that a task needs to be processed by forwarding the task, represented as task information in a common message format, to the blending engine, as shown in block 164. In one embodiment, the blending engine maintains a set of queues in the blending database, one for each media switch. In this way, the blending database maintains a snapshot of the workload of the media switches of the contact center. Using this snapshot, combined with agent availability data and agent skill data of the agents, workflows determine the most suitable agent for the task at the particular moment in time. When a task is queued on a media switch, it is added to the blending database with a status of AVAILABLE, as shown in block 166. A workflow is then executed to determine whether a suitable agent is available to service the task, as shown in block 168.

The emphasis in "task queued" workflows is finding agents that are available to handle the task. In one embodiment, if multiple agents are available, the task is routed to the agent that has waited longest since their previous task. In another embodiment, assignment of tasks is based on that agent's relative skill in a particular medium. For example, in such an embodiment, if an email message is to be serviced, the email message is routed to the agent with the highest email-handling skill level. In yet another embodiment, a task is routed based on consideration of the business area to

which the contact relates so that the agent with the highest skill in that business area is assigned the task. The criteria and rules for these decisions are dependent on the information available to the workflow. In various embodiments, the blending database maintains pertinent data about each agent that may be important in determining which agent should be assigned a particular task. In these embodiments, the business rules may contemplate the following: the agent's fluency in a language that may be required to service the task is considered, the number of hours the agent has been at work without a break, the agent's expertise in a product area, the agent's proficiency in using particular kinds of media, etc.

In one embodiment, the business rules of workflows may be customizable and exist in multiple varying embodiments. Workflows can be very complex, with decisions being made based on many different sources of information. For example, in one embodiment, workflows interrogate a customer database in addition to the blending database to determine whether the caller is a high-value, VIP customer, or a less important customer, and route the call accordingly. In another example, another embodiment provides for workflows communicating with older systems, known as legacy systems, to obtain information based on the business rules built into the older system. In such an embodiment, the investment already made in building the older business rules is not lost, and the business rules are not duplicated in a newly added component. Any of the workflow server's capabilities can be leveraged in this way to provide highly complex decision-making.

In yet another embodiment, to allow for workflows to be run at high-volumes, one for each task queued, where intricate workflows include complex database queries, slow networking links and large data sets, throughput is increased and latency is reduced by database caching and/or database replication, i.e., redundancy or multiple copies of the same database on multiple computers. In one embodiment, when a task is dequeued or terminated, it is removed from the blending database, but no workflow is invoked. In another embodiment,

when a task is dequeued or terminated, an appropriate workflow such as, for example, a "task dequeued" workflow may be executed.

Using the above and other criteria, the workflow determines if a suitable agent is available to service the task, as shown in block 168. The workflow server then advises the blending engine of its decision as to which, if any, suitable agent is available to service the task, as shown in block 170. If a suitable agent is not available, the task remains in the blending database, with status AVAILABLE, as shown in blocks 170 and 172. The task is later assigned either when an agent requests more work, if the media switch assigns the task to a non-blended agent, or if another custom workflow assigns the task, such as when a certain condition (like call volume exceeding a defined threshold) exists. If a suitable agent is available, the blending engine sends the details of that task to the agent's desktop, as shown in blocks 170 and 174. In one embodiment, not shown, the desktop application receives the task information and determines whether the agent accepts the task. In this embodiment, the action in blocks 52, 54, 58 and 46 of **Figure 2** may be executed. In this embodiment, if a suitable agent is available that accepts the task, the desktop application uses information about the task to determine what media switch and/or medium corresponds to the task, as shown in block 175. The relevant API is then used to request the task from the particular media switch, as shown in block 176. While this process is carried out, the agent's blending status is set to RESERVED to signal to other workflows that the agent is not available to be assigned tasks.

There is a possibility that the task being requested may no longer be queued when the desktop application requests it. This may occur, for example, when the media switch assigns the task to a non-blended agent, or, in the case of voice calls, when the caller hangs up. Generally, this scenario is catered for by dequeued / terminated messages being sent by the media switch which causes the blending engine to remove the task from the blending queue. However, the possibility that the sequencing of events may lead to a task being requested when the task is no longer available means that the desktop must be able to handle this

scenario. At this point, the desktop application must determine whether the task has been received, as shown in block 178, by, in one embodiment, timing out, or, in another embodiment, processing a task not available notification. If the task an agent was assigned no longer exists, the desktop application requests another task, in one embodiment, by setting the agent status to AVAILABLE as shown in block 180.

If the task is received by the desktop application, as shown in block 178, the desktop application will inform the agent manager via the desktop helper that this has occurred, as shown in block 182. In addition, the agent message handler will instruct the database manager to update the agent's status to UNAVAILABLE, as shown in block 184. The media switch also generates a "task dequeued" message at this point, causing that task to be removed from the blending database, as shown in blocks 186 and 188.

In yet another embodiment, the "task queued" workflow may assign tasks to electronic devices via software agents in addition to human agents. In such an embodiment, automatic email, and/or automatic voice generation or play back may provide information requested by the task. For example, the task may be an email or telephonic request for a current balance or information about the last three transactions or last three checks that cleared. Such information can easily automatically be retrieved and provided via email or text to voice synthesis via telephone. Similarly, driving directions to a facility or store policies may be pre-recorded in text or voice and automatically be sent as email or played back via a telephone.

2. Agent Available Workflow

Figure 5 illustrates the flow of actions taken by a blending system upon an agent becoming available. An "agent available" workflow is invoked when an agent finishes handling a task and indicates that the agent is ready for another task. In one embodiment, a notification that an agent is ready for another task is agent-initiated. In this embodiment, the agent presses a button on a desktop application to indicate that the agent is ready for another task, causing a message

stating that the agent is available to be sent to the blending engine, as shown in block 200. In another embodiment, a notification that an agent is ready for another task is automatic. In this embodiment, when the agent finishes with a task, the application used to carry out that task recognizes that the task has been completed, and communicates with the blending engine, sending a message that the agent is available to service another task, as shown in block 202. In one embodiment, changing the agent's status from "available if needed" to "available" triggers a communication to the blending engine that the agent is available to service a task.

After receipt of an agent available message, as shown in block 204, the blending engine sends a message to the blending database placing the agent in a state of RESERVED, as shown in block 206, and the blending engine then invokes the agent available workflow by messaging the workflow server, as shown in block 208. The agent is put in a state of RESERVED in order to avoid the situation where a separate "task queued" workflow selects that agent for its task while the workflow is executing attempting to find the agent a new task. The workflow server then proceeds to find a task suitable for the agent by querying the blending database. If no suitable pending task is found, the agent state will be set to AVAILABLE, as shown in blocks 210 and 212.

If a pending task is found for the agent, the blending database entry for the task is marked as RESERVED, as shown in blocks 210 and 214. The blending database entry for the task is marked as RESERVED in order that another "agent available" workflow does not assign the task to another agent. The blending engine then sends information about the task to the agent, as shown in block 216. The desktop application uses the task information received from the blending engine to retrieve the task by extracting the task medium from the task information, as shown in block 218. The agent then requests the task from the corresponding media switch, as shown in block 220. If the task no longer exists, the desktop agent requests another task, as shown in blocks 222 and 224. If the desktop component succeeds in retrieving the task from the media switch, as

shown in block 222, the agent manager component of the blending engine is informed that the task has been successfully assigned, as shown in block 226, and the agent message handler informs the database manager to update the agent's status to UNAVAILABLE, as shown in block 228. As discussed above, the removal of the task from the media switch causes a "task dequeued" message to be generated by the media switch and sent to the blending engine, as shown in block 230. This message causes the task to be removed from the blending database.

In another embodiment, when an agent requests more work, the agent may request that a particular medium be excluded. This embodiment is referred to as the empowered agent embodiment. For example, in this embodiment, an the agent may set certain parameters specify that, despite call levels being high, the agent will not accept any calls, preferring to deal with email messages exclusively.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes can be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. Therefore, the scope of the invention should be limited only by the appended claims.